

EV205822984

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

Vision-Based Document Segmentation

Inventor(s):
Ji-Rong Wen
Shipeng Yu
Deng Cai
Wei-Ying Ma

ATTORNEY'S DOCKET NO. MS1-1616US

1 **TECHNICAL FIELD**

2 This invention relates to partitioning documents, and particularly to vision-
3 based document segmentation.

4
5 **BACKGROUND**

6 People have access to a great deal of information. However, finding the
7 particular information they desire in any given situation can be very difficult. For
8 example, a large amount of information is accessible to people over the Internet in
9 the form of web pages. The number of such web pages can be on the order of
10 millions or more. Additionally, the web pages available are constantly changing,
11 with some pages being added, others being deleted, and others being modified.

12 Thus, when someone desires to find some information, such as the answer
13 to a question, the ability to extract particular information from this large
14 information source becomes very important. Processes and techniques have been
15 developed to allow users to search for information over the Internet, and are
16 commonly made available to the user in the form of search engines. However, the
17 accuracy of such search engines can be lacking due in large part to the extremely
18 broad range of content on web pages that are searched. For example, some web
19 pages include copyright and other business-related notices, and some web pages
20 include advertisements. Such business-related and advertising data is not always
21 relevant to the underlying content of the web page, and thus can reduce the
22 accuracy of the searching process if it is considered. By way of another example,
23 different web pages can vary greatly in length, and some may include multiple
24 topics while others contain a single topic.

1 These characteristics of web pages can reduce the accuracy of search
2 processes. Thus, it would be beneficial to have a way to increase the accuracy of
3 searching documents.

4 5 **SUMMARY**

6 Vision-based document segmentation is described herein.

7 In accordance with one aspect, one or more portions of semantic content of
8 a document are identified. The one or more portions are identified by identifying
9 a plurality of visual blocks in the document, and detecting one or more separators
10 between the visual blocks of the plurality of visual blocks. A content structure for
11 the document is constructed based at least in part on the plurality of visual blocks
12 and the one or more separators, and the content structure identifies the different
13 visual blocks as different portions of semantic content of the document.

14 In accordance with other aspects, the content structure obtained using the
15 vision-based document segmentation is used during document retrieval.

16 17 **BRIEF DESCRIPTION OF THE DRAWINGS**

18 The same numbers are used throughout the document to reference like
19 components and/or features.

20 Fig. 1 is a block diagram illustrating an example system using vision-based
21 document segmentation.

22 Fig. 2 is a flowchart illustrating an example process for performing vision-
23 based document segmentation.

24 Fig. 3 is a flowchart illustrating an example process for performing visual
25 block identification.

1 Fig. 4 is a flowchart illustrating an example process for performing visual
2 separator detection.

3 Figs. 5a, 5b, 5c, 5d, 5e, and 5f illustrate an example of detecting horizontal
4 separators.

5 Fig. 6 is a flowchart illustrating an example process for performing content
6 structure construction.

7 Fig. 7 illustrates an example of generating virtual blocks and a content
8 structure tree.

9 Fig. 8 is a flowchart illustrating an example process for performing
10 document retrieval using vision-based document segmentation.

11 Fig. 9 is a flowchart illustrating another example process for performing
12 document retrieval using vision-based document segmentation.

13 Fig. 10 is a flowchart illustrating another example process for performing
14 document retrieval using vision-based document segmentation.

15 Fig. 11 illustrates an example of a general computer environment, which
16 can be used to implement the techniques described herein.

17
18 **DETAILED DESCRIPTION**

19 Vision-based document segmentation is described herein. The vision-based
20 document segmentation identifies, based on the visual appearance of a document,
21 multiple portions of the document that include the semantic content of that
22 document. The vision-based document segmentation can be used in a variety of
23 different manners. For example, the segmentation can be used when searching for
24 documents, so that the search results are based on the semantic content portions of
25 the document.

1 The discussions herein refer to documents and models used to describe the
2 structures of the documents. Documents can be in any of a variety of formats,
3 such as in accordance with a Standard Generalized Markup Language (SGML),
4 such as the Extensible Markup Language (XML) format or the HyperText Markup
5 Language (HTML) format. In certain embodiments, these documents are web
6 pages in the HTML format. The models discussed herein can be any of a variety
7 of models that describe the structure of a document. In certain embodiments, the
8 model used is a Document Object Model (DOM). The Document Object Model is
9 a tree-structured representation of a document, also referred to as a DOM tree. In
10 many of the discussions herein, the documents are described as being in the
11 HTML format (e.g., web pages), the model is described as a DOM tree, and each
12 HTML tag of the document is represented by a node in the DOM tree (the DOM
13 tree may also include additional nodes, such as #text or #comment nodes, which
14 may not represent an HTML tag). However, it is to be appreciated that the vision-
15 based document segmentation can also be used with these other documents and/or
16 models.

17 Fig. 1 is a block diagram illustrating an example system 100 using the
18 vision-based document segmentation. System 100 includes a vision-based
19 document segmentation module 102 having a visual block extractor 104, a visual
20 separator detector 106, and a content structure constructor 108. A document
21 description is accessible to vision-based document segmentation module 102.
22 This document description is a model that describes the structure of the document,
23 such as a DOM tree. The document description may be generated by another
24 component (not shown) and made accessible to module 102, or alternatively the
25

1 document itself may be accessible to module 102 and module 102 may generate
2 the document description.

3 Visual block extractor 104 identifies, based on the document description,
4 multiple visual blocks of the document. This identification (also referred to herein
5 as extraction) of the visual blocks is based on visual cues within the document,
6 such as font sizes and/or types, colors of fonts and/or background information,
7 HTML tag type, and so forth. The identified blocks are regions of the document.
8 The identified blocks are output by extractor 104 and made available to visual
9 separator detector 106.

10 Visual separator detector 106 detects separators between different ones of
11 the identified blocks. These separators can take a variety of different forms, such
12 as lines in the document, blank space in the document, different background colors
13 for different blocks, and so forth. In certain embodiments, separators in the
14 horizontal and/or vertical direction in the document are detected. These detected
15 separators are made available to content structure constructor 108.

16 Content structure constructor 108 generates a content structure for the
17 document based on the blocks identified by extractor 104 as well as the separators
18 detected by detector 106. The content structure is a set of one or more portions of
19 the document that represents the semantic content of a document. The content
20 structure is output by module 102.

21 The content structure of the document can then be used in a variety of
22 manners. In certain embodiments, system 100 includes an optional document
23 retrieval module 110. Document retrieval module 110 uses the content structure
24 from vision-based segmentation module 102 in determining which documents to
25 return in response to a query. For example, when a user enters one or more search

1 terms when searching for web pages, the content structure can be used in
2 determining which web pages satisfy the criteria.

3 The manner in which the content structure is used during document
4 retrieval can vary. In some embodiments, the content structure is used to rank the
5 documents that are retrieved in response to the query. Additionally, in some
6 embodiments the content structure is used as a basis for expanding the search
7 criteria. These uses are discussed in more detail below.

8 It should be noted that the vision-based document segmentation described
9 herein makes use of the way a document would look when it is displayed. The
10 vision-based document segmentation does not require that a document actually be
11 displayed, and the vision-based document segmentation also does not require that
12 the user actually see a displayed document.

13 Fig. 2 is a flowchart illustrating an example process 140 for performing
14 vision-based document segmentation. Process 140 is implemented by vision-
15 based document segmentation module 102 of Fig. 1, and may be performed in
16 software, firmware, hardware, or combinations thereof.

17 Initially, a document description is accessed (act 142). This document
18 description is, for example, a DOM tree for an HTML web page. With the
19 document description, one or more visual blocks in the document are identified
20 (act 144). Separators between the identified visual blocks are detected (act 146),
21 and a content structure for the document is constructed based on the identified
22 visual blocks and the detected separators (act 148). A check is then made as to
23 whether the content structure satisfies a granularity requirement (act 150). This
24 granularity requirement refers to a degree of coherence (DoC) for the visual blocks
25 of the content structure, which is a measure of how coherent each of the visual

1 blocks is. If the granularity requirement is not satisfied then process 140 returns to
2 act 144, where new visual blocks are identified within the portions of the content
3 structure that did not satisfy the granularity requirement. Once the granularity
4 requirement is satisfied, the content structure for the document is output (act 152).
5 Each pass through the combination of acts 144, 146, and 148 (that is, visual blocks
6 are identified, separators between the identified blocks are detected, and content
7 structure based on the identified visual blocks and detected separators is
8 constructed) is referred to herein as a "round".

9 Thus, it can be seen that the vision-based document segmentation employs
10 a top-down approach. Generally, an initial set of visual blocks is identified and
11 made available for visual separator detection and content structure construction as
12 discussed below. Within these visual blocks, additional visual blocks are
13 identified in subsequent rounds as appropriate (that is, additional visual blocks are
14 identified in subsequent rounds for each visual block that does not satisfy the
15 granularity requirement).

16 As discussed herein, the documents being used are assumed to be
17 rectangular in shape when displayed. Thus, some sizes, dimensions, areas, and so
18 forth are described herein with reference to a rectangular or Cartesian coordinate
19 system (e.g., an X,Y-coordinate system). It is to be appreciated that these are only
20 examples, and that the rectangular shapes and the coordinate systems can be
21 different. In some instances, if non-rectangular shapes are used they are converted
22 to rectangular shapes prior to performing the vision-based document segmentation.
23 In other instances, the processes and techniques described herein are adapted to
24 use these different shapes and/or coordinate systems.
25

Visual Block Identification

Visual block identification, also referred to herein as visual block extraction, identifies different regions of the document based on visual cues. An indication of these different regions is then output and used in the visual separator detection, as discussed in more detail below. Visual block identification (e.g., in act 144 of Fig. 2 or implemented by visual block extractor 104 of Fig. 1) can be performed as follows.

Generally, every node in the DOM tree for a document can represent a visual block. Some large nodes (e.g., having <TABLE> or <P> tags) are typically used only for organization purpose and are not appropriate to represent a single visual block. Thus, such large nodes are further divided and replaced by their children nodes. On the other hand, because of the potentially large number of leaf nodes, the visual blocks should not be initially identified as being every leaf node in the DOM tree (although, due to the top-down nature of the vision-based document segmentation, some of these leaf nodes may eventually be identified as visual blocks).

Fig. 3 is a flowchart illustrating an example process 170 for performing visual block identification. Process 170 illustrates act 144 of Fig. 2 in additional detail. Process 170 may be performed in software, firmware, hardware, or combinations thereof.

Initially, a node from a group of candidate nodes of the DOM tree is selected (act 172). This group of candidate nodes refers to nodes of the DOM tree that are potentially visual blocks in the current round. The candidate nodes can change while process 170 is being performed, as discussed in more detail below. In certain embodiments, the group of candidate nodes is initially a single node, the

1 single node being the root node of the DOM tree in the first round or the top node
2 of a subtree of the DOM tree in subsequent rounds (e.g., the top node of a part of
3 the DOM tree corresponding to a visual block that did not satisfy the granularity
4 requirement). Alternatively, the initial group of candidate nodes may be different,
5 such as the nodes one level below the root node in the DOM tree or the nodes one
6 level below the top node of a subtree of the DOM tree.

7 A check is then made as to whether the selected node can be divided (act
8 174). If the selected node can be divided, then each child node of the selected
9 node is added to the group of candidate nodes (act 176), and the selected node is
10 removed from the group of candidate nodes (act 178). Process 170 then returns to
11 act 172 and selects another node from the group of candidate nodes.

12 Returning to act 174, if the selected node cannot be divided, then the
13 selected node represents a visual block (act 180), and is added to a pool of visual
14 blocks being identified by process 170. A degree of coherence (DoC) value is set
15 for the visual block (act 182), as discussed in more detail below, and the selected
16 node is removed from the group of candidate nodes (act 184). A check is also
17 made as to whether there are additional nodes in the group of candidate nodes (act
18 186). If there are additional nodes in the group of candidate nodes, then process
19 170 returns to act 172, where one of the nodes is selected from the group of
20 candidate nodes. If there are not additional nodes in the group of candidate nodes,
21 then the visual block identification process 170 is finished (act 188), with the
22 visual blocks selected in act 180 being the visual blocks identified by process 170.
23 This pool of visual blocks identified by process 170 can then be used in the visual
24 separator detection, discussed in more detail below.
25

1 The determination of whether a particular node can be divided (in act 174)
2 is made based on one or more rules. These rules make use of one or more cues or
3 information regarding the node and possibly one or more other nodes in the DOM
4 tree. Examples of such cues or information include a tag cue, a color cue, a text
5 cue, and a size cue. Examples of the rules that can be formulated based on these
6 cues follow. In certain embodiments, a default rule (e.g., that the node can be
7 divided, or alternatively that the node cannot be divided) may also be included.
8 *The default rule is used if none of the other rules are applicable.*

9 Top Block Rule: The top block rule is based on the location of the node in
10 the tree being evaluated. The top block rule states that if the node is the top node
11 in the tree being evaluated (e.g., the root node of the DOM tree in the first round),
12 then the node can be divided.

13 Tag Rule: The tag rule is based on the HTML tag of the node. Some
14 HTML tags, such as the <HR> tag, are often used to separate different topics from
15 a visual perspective. The tag rule states that if the HTML tag of one of the
16 children of the node is equal to one of a set of particular tags (e.g., the HTML tag
17 is the <HR> tag), then the node can be divided.

18 Color Rule: The color rule is based on the background color of the node as
19 well as the background color(s) of the children of the node. The color rule states
20 that if the background color of the node is different from the background color of
21 at least one of its children nodes, then the node can be divided.

22 Text and Size Rules: Several text and size rules can be used. Reference is
23 made in some of these rules to the height, width, and/or size of a node. The height
24 of a node refers to the visual height of the block (e.g., in pixels using an X,Y-
25 coordinate system) represented by the node when the block is displayed, the width

1 of a node refers to the visual width of the block (e.g., in pixels using an X,Y-
2 coordinate system) represented by the node when the block is displayed, and the
3 area of a node refers to the display area (e.g., in pixels using an X,Y-coordinate
4 system) occupied by the block represented by the node when it is displayed (e.g.,
5 the product of the width and height of the node in the case of a rectangular block).
6 These sizes can be readily determined based on the position location information
7 included in the nodes of the DOM tree.

8 Some of these rules (as well as discussions elsewhere in this description)
9 refer to a valid node, a block node, a text node, a virtual text node, and/or a
10 navigation node. A valid node refers to a node that is visible when the HTML
11 document is displayed (e.g., neither the height nor the width is equal to zero). A
12 block node refers to a node that has some tag other than <A>, , ,
13 <HR>, <I>, <P>, , or <TEXT>. A text node refers to a node
14 that only contains free text. A virtual text node refers to a node that is not a block
15 node and only has text nodes as children (or child). A navigation node refers to a
16 node having a height that is more than twice its width, and having a width smaller
17 than 200. In certain implementations, rules regarding navigation nodes are used in
18 the first round, but not in subsequent rounds.

19 The following text and size cue based rules can be used for nodes having
20 general tags (that is, for all HTML tags except for the <P> tag, the <TABLE> tag,
21 the <TD> tag, and the tag):

- 22 • if the node has no valid children then the node cannot be divided and
23 the node is removed from the group of candidate nodes;

- 1 if the node has only one valid child and the child is not a text node,
2 then trace into the child (remove the node from the group of
3 candidate nodes and replace it with the child node);
- 4 • if the node's size is at least three times greater than the sum of its
5 children's sizes, then the node can be divided;
 - 6 • if all of the children of the node are text nodes or virtual text nodes,
7 then the node cannot be divided;
 - 8 • if the node has at least one text node child or at least one virtual text
9 node child, and the node's width or height is smaller than a threshold
10 (e.g., 150), then the node cannot be divided;
 - 11 • if the node is a navigation node, then the node cannot be divided;
 - 12 • if the node has at least one text node child or at least one virtual text
13 node child, and further does not have a block node child, then the
14 node cannot be divided;
 - 15 • if the node has more than two successive
 children, then the
16 node can be divided; and
 - 17 • otherwise, the node can be divided.

18 The following text and size cue based rules can be used for nodes having
19 <P> HTML tags:

- 20 • if the node has no valid children then the node cannot be divided and
21 the node is removed from the group of candidate nodes;
 - 22 • if the node has only one valid child and the child is not a text node,
23 then trace into the child (remove the node from the group of
24 candidate nodes and replace it with the child node);
- 25

1 if all of the children of the node are text nodes or virtual text nodes,
2 then the node cannot be divided;

- 3 • if the sum of all the children's size are greater than this node's size,
4 and there is at least one child that is a block node, then the node can
5 be divided.
- 6 • if the node has at least one text node child or at least one virtual text
7 node child, and the node's width or height is smaller than a threshold
8 (e.g., 150), then the node cannot be divided;
- 9 • if the node has no block node child, then the node cannot be divided.
- 10 • if all the children of the node are smaller than a threshold (e.g.,
11 Width < 150 and Height < 100), then the node cannot be divided;
- 12 • otherwise, the node can be divided.

13 The following text and size cue based rules can be used for nodes having
14 <TABLE> HTML tags:

- 15 • if the node has no valid children then the node cannot be divided and
16 the node is removed from the group of candidate nodes;
- 17 • if the node has only one valid child and the child is not a text node,
18 then trace into the child (remove the node from the group of
19 candidate nodes and replace it with the child node);
- 20 • if the size of the biggest child node of the node is smaller than a
21 threshold (e.g., 250*200), then the node cannot be divided;
- 22 • if all the children of the node are smaller than a threshold (e.g.,
23 Width < 150 and Height < 100), then the node cannot be divided;
- 24
- 25

1 if all the valid node children of the node are bigger than a threshold
2 (e.g., Width > 150 and Height > 100), then the node cannot be
3 divided.

- 4 • if some (greater than 0) valid node children of the node are bigger
5 than a threshold (e.g., Width > 150 and Height > 100) and some
6 (greater than 0) valid node children of the node are smaller than a
7 threshold (e.g., Width < 150 and Height < 100), then the node cannot
8 be divided.
- 9 • otherwise, the node can be divided.

10 The following text and size cue based rules can be used for nodes having
11 <TD> HTML tags:

- 12 • if the node has no valid children then the node cannot be divided and
13 the node is removed from the group of candidate nodes;
- 14 • if the node has only one valid child and the child is not a text node,
15 then trace into the child (remove the node from the group of
16 candidate nodes and replace it with the child node);
- 17 • otherwise, the node cannot be divided.

18 The following text and size cue based rules can be used for nodes having
19 HTML tags:

- 20 • if the node has no valid children then the node cannot be divided and
21 the node is removed from the group of candidate nodes;
- 22 • if the node has only one valid child and the child is not a text node,
23 then trace into the child (remove the node from the group of
24 candidate nodes and replace it with the child node);

1 if the node does not have a block node child, then the node cannot be
2 divided;

- 3 • if all the children nodes of the node have HTML tags, then the
4 node cannot be divided.
- 5 • if all the children nodes of the node are smaller than a threshold
6 (e.g., Width < 150 and Height < 100), then the node cannot be
7 divided.
- 8 • otherwise, the node can be divided.

9 In visual block identification, the degree of coherence (DoC) value is a
10 measure assigned to visual blocks extracted from the DOM tree. A different
11 degree of coherence (DoC) value may also be generated for virtual blocks during
12 the content structure construction, as discussed in more detail below. The DoC
13 value of a visual block (or virtual block) is a measure of how coherent the visual
14 block (or virtual block) is.

15 The DoC value assigned to a block during visual block identification can be
16 assigned in a variety of different manners. In certain embodiments, the DoC value
17 of a visual block is assigned based on its corresponding node in the DOM tree (the
18 node that represents the visual block), and can vary based on the HTML tag of the
19 node. The following is a set of example rules that can be followed in assigning
20 DoC values during the visual block identification. In the following examples,
21 DoC values are integers ranging from 1 to 10, although alternatively different
22 ranges (e.g., made up of integers, real numbers, etc.) could be used. Situations
23 may arise where multiple ones of the following rules may apply to a particular
24 block, and could result in multiple DoC values for the block. Such situations can
25 be resolved in different manners, such as selecting the largest DoC value assigned

1 by one of the rules, selecting the smallest DoC value assigned by one of the rules,
2 averaging the DoC values assigned by the rules, calculating a weighted average of
3 the DoC values assigned by the rules, and so forth.

4 The following example rules are used to assign DoC values to visual blocks
5 having general HTML tags (that is, to all HTML tags except for the <P> tag, the
6 <TABLE> tag, the <TD> tag, and the tag):

- 7 • if all of the children of the node are text nodes or virtual text nodes,
8 then set the DoC value for the block to 10;
- 9 • if the node has at least one text node child or at least one virtual text
10 node child, and the node's width or height is smaller than a threshold
11 (e.g., 150), then set the DoC value for the block to 8;
- 12 • if the node is a navigation node (e.g., in the first round only, a node
13 having a height that is more than twice its width, and having a width
14 smaller than 200; in subsequent rounds, this rule is not applied), then
15 set the DoC value for the block to 7;
- 16 • if the node has at least one text node child or at least one virtual text
17 node child, and further does not have a block node child, then set the
18 DoC value for the block based on the node size as follows:
 - 19 ○ if both the width and height of the node are less than a first
20 threshold value (e.g., 100), then set the DoC value for the
21 block to 8;
 - 22 ○ if the height of the node is less than a second threshold (e.g.,
23 200) or the width is less than a third threshold (e.g., 400),
24 then set the DoC value for the block to 7; and
25

- if the area of the node (e.g., the product of the height and width of the node) is less than a fourth threshold (e.g., 100,000), then set the DoC value for the block to 6; and
- if none of the above rules applies to a block having a general HTML tag, then set the DoC value for the block to 5.

The following example rules are used to assign DoC values to visual blocks having <P> HTML tags:

- if all the children of the node are text nodes or virtual text nodes, then set the DoC value for the block to 10;
- if the node has at least one text node child or at least one virtual text node child, and the node's width or height is smaller than a threshold (e.g., 150), then set the DoC value for the block to 8;
- if the node does not have a block node child, then set the DoC value for the block to 7; and
- if all of the block node children of the node are smaller than a threshold (e.g., Width < 150 and Height < 100), then:
 - if the node has at least one text node child or at least one virtual text node child, then set the DoC value for the block to 7; and
 - otherwise, set the DoC value for the block to 6.

The following example rules are used to assign DoC values to visual blocks having <TABLE> HTML tags:

- if the size of the biggest child node of the node is smaller than a threshold (e.g., 250*200), then set the DoC value for the block to 8;

1 if all the children of the node are smaller than a threshold (e.g.,
2 Width < 150 and Height < 100), then set the DoC value for the block
3 to 7; and

- 4 • if all the valid node children of the node are bigger than a threshold
5 (e.g., Width > 150 and Height > 100), then set the DoC value for the
6 block to 6.

7 The following example rules are used to assign DoC values to visual blocks
8 having <TD> HTML tags:

- 9 • if all the children of the node are text nodes or virtual text nodes,
10 then set the DoC value for the block to 10;
- 11 • if the node has at least one text node child or at least one virtual text
12 node child, and the node's width or height is smaller than a threshold
13 (e.g., 150), then set the DoC value for the block to 8;
- 14 • if the node does not have a block node child, then set the DoC value
15 for the block to 7;
- 16 • if all of the block node children of the node are smaller than a
17 threshold (e.g., Width < 150 and Height < 100), then set the DoC
18 value for the block to 7; and
- 19 • if the size of the biggest child node of the node is smaller than a
20 threshold (e.g., 250*200), then set the DoC value for the block to 7.

21 The following example rules are used to assign DoC values to visual blocks
22 having HTML tags:

- 23 • if the node does not have a block node child, then set the DoC value
24 for the block to 8;

- if all the children nodes of the node have HTML tags, then set the DoC value for the block to 8; and
- if all the children nodes of the node are smaller than a threshold (e.g., Width < 150 and Height < 100), then set the DoC for the block to 8.

These rules and an example of process 170 of Fig. 3 can be implemented, for example, using an algorithm having a DivideDomtree procedure or function (shown in Table I below), and a Dividable procedure or function (shown in Table II below). In this algorithm, the value *pRoot* refers to the currently selected node, the value *nLevel* refers to the level of the currently selected node in the DOM tree, the pool refers to the group of visual blocks identified by the algorithm, the Top Block refers to the top node of the DOM tree or subtree being evaluated in this round (e.g., in the first round, the Top Block refers to the root node of the DOM tree), and child refers to a child of the currently selected node. The special routines and heuristic rules of the Dividable procedure or function are described above. Each time the DivideDomtree procedure or function is invoked, it is passed a node of the DOM tree as the value of *pRoot*, and a value of the level of that node in the DOM tree as the value of *nLevel*.

Table I

```
Algorithm DivideDomtree(pRoot, nLevel)
{
  IF (Dividable(pRoot, nLevel) == TRUE){
    FOR EACH child OF pRoot {
      DivideDomtree(child, nLevel);
    }
  } ELSE {
    Put the sub-tree (pRoot) into the pool as a block;
  }
}
```

Table II

```
Algorithm Dividable(pRoot, nLevel)
{
  IF (pRoot is the Top Block){
    RETURN TRUE;
  } ELSE {
    Special routines for TABLE, TD, P, UL;
    Heuristic rules for general tags;
  }
}
```

Visual Separator Detection

Visual separator detection detects separators between different ones of the blocks identified by the visual block identification. These separators can take a variety of different forms, such as lines in the document, blank space in the document, different background colors for different blocks, and so forth. In certain embodiments, separators in the horizontal and/or vertical direction in the document are detected. Visual separator detection (e.g., in act 146 of Fig. 2 or

1 implemented by visual separator detector 106 of Fig. 1) can be performed as
2 follows.

3 The detected separators are horizontal or vertical regions in the document
4 that do not visually intersect with any blocks in the pool of blocks identified by the
5 visual block identification. These separators are used as indicators for
6 discriminating different semantics within the document. A separator is
7 represented by a 2-tuple (P_s , P_e), where P_s is the start pixel of the separator and P_e
8 is the end pixel of the separator. Each pixel P_s and P_e is identified as an X,Y-
9 coordinate pair, and a rectangular region can be defined by setting the P_s and P_e
10 pixels as opposite corners of the region (e.g., lower left and upper right corners,
11 upper left and lower right corners, etc.). The width of the separator is calculated
12 as the difference (e.g., in the X direction) between P_s and P_e .

13 Reference is made to horizontal and vertical directions. Typically, in an
14 X,Y-coordinate system, the horizontal direction refers to the X-axis while the
15 vertical direction refers to the Y-axis. However, other orientations for the
16 horizontal and vertical directions can alternatively be used.

17 Fig. 4 is a flowchart illustrating an example process 200 for performing
18 visual separator detection. Process 200 illustrates act 146 of Fig. 2 in additional
19 detail. Process 200 may be performed in software, firmware, hardware, or
20 combinations thereof.

21 Initially, a separator list is initialized with one separator (act 202). This one
22 separator includes all regions of the display area for the document that may
23 potentially be a separator(s). In certain embodiments, this initial separator
24 includes the entire area of the document, or alternatively this initial separator may
25

1 be a single rectangular region that includes all of the blocks in the pool of visual
2 blocks identified by the visual block identification.

3 A block from the pool of identified blocks is then selected (act 204).
4 Blocks can be selected from the pool in any of a variety of manners (e.g.,
5 randomly, in the order they were added to the pool, by increasing or decreasing
6 size, and so forth). A check is then made as to whether the selected block is
7 contained in a separator of the separator list (act 206). A block is considered to be
8 contained in a separator if the entire area of a block is included within the area of
9 the separator. If the selected block is contained in a separator, then the separator is
10 split into two separators (act 208). This split could be accomplished in different
11 manners, such as by removing the separator that contains the block from the
12 separator list and adding two new separators to the separator list, modifying the P_s
13 or P_e pixels of the separator and adding a new separator, and so forth. The
14 splitting of the separator results in what was previously a single separator
15 becoming two smaller separators on either side of the block. When detecting
16 horizontal separators, the two smaller separators would be above and below the
17 block, and when detecting vertical separators, the two smaller separators would be
18 to the left and to the right of the block.

19 After the separator is split in act 208, or if the selected block is not
20 contained in a separator, a check is made as to whether the selected block crosses a
21 separator (act 210). When detecting horizontal separators, a block is considered to
22 cross a separator if the area of the block intersects a portion of the separator but
23 less than the entire height of the separator. When detecting vertical separators, a
24 block is considered to cross a separator if the area of the block intersects a portion
25 of the separator but less than the entire width of the separator.

1 If the selected block crosses a separator, then the parameters of the
2 separator are updated so that the block no longer crosses the separator (act 212).
3 This updating of the parameters refers to modifying the P_s and/or P_e pixels of the
4 separator so that the block no longer crosses the separator.

5 After the separator parameters are updated in act 212, or if the selected
6 block does not cross a separator, a check is made as to whether the selected block
7 covers a separator (act 214). When detecting horizontal separators, a block is
8 considered to cover a separator if the area of the block intersects the entire height
9 of at least part of the separator. When detecting vertical separators, a block is
10 considered to cover a separator if the area of the block intersects the entire width
11 of at least part of the separator.

12 If the selected block covers a separator, then the separator is removed from
13 the separator list (act 216). After the separator is removed from the separator list
14 in act 216, or if the selected block does not cover a separator, a check is made as to
15 whether there are any additional blocks in the pool of identified blocks that have
16 not yet been selected (act 218). If there are such blocks in the pool of blocks, then
17 process 200 returns to select one of these remaining blocks in act 204. However,
18 if there are no such blocks in the pool of blocks, then the four separators at the
19 borders of the display (if such separators are present) are removed from the
20 separator list (act 220).

21 The separator detection is performed to detect both horizontal separators
22 and vertical separators. Horizontal separators could be detected first and then
23 vertical separators detected, or vertical separators could be detected first and then
24 horizontal separators detected, or alternatively the horizontal and vertical
25 separators could be detected concurrently.

1 In certain embodiments, acts 202 through 218 of Fig. 4 are performed for
2 both horizontal and vertical separators. Thus, two different separator lists can be
3 generated. These two separator lists are combined in any of a variety of manners,
4 such as taking the union of the two separator lists. For example, a final separator
5 list may be generated that includes every separator in one of the two lists.

6 Thus, as can be seen from Fig. 4, the visual separator detection begins with
7 a separator list includes one or more possible separator(s), and the list of possible
8 separators is changed until, when the detection process is finished, the list includes
9 the detected separators. The separator list is changed to add new separators,
10 remove separators, and/or modify the parameters of separators based on whether
11 the visual blocks overlap the separators as well as how they overlap the separators
12 (e.g., the block is contained in a separator, the block crosses a separator, or the
13 block covers a separator).

14 It should also be noted that the separators detected by process 200 can take
15 any of a variety of different forms when the document is displayed. For example,
16 the separators may be blank space in the document, one or more separator lines
17 drawn in the document, images or other graphics in the documents, portions of the
18 document shaded different colors, and so forth.

19 By way of example, detecting horizontal separators can be seen in the
20 example illustrated in Figs. 5a – 5f. Assume that the visual block identification
21 identifies four visual blocks 240, 242, 244, and 246 in a document 248 of Fig. 5a.
22 The visual blocks 240, 242, 244, and 246 are illustrated with diagonal cross-
23 hatching. Initially, only one separator 250 is in the separator list as illustrated in
24 Fig. 5b. The separators are illustrated in Figs. 5b through 5f with vertical lines.
25 The separator 250 includes all of the visual blocks 240, 242, 244, and 246.

1 Further assume that visual block 240 is the first block selected from the pool of
2 visual blocks, visual block 242 is the second block selected from the pool of visual
3 blocks, visual block 244 is the third block selected from the pool of visual blocks,
4 and visual block 246 is the fourth block selected from the pool of visual blocks.

5 Visual block 240 is contained in separator 250, so separator 250 is split into
6 two separators 252 and 254 as illustrated in Fig. 5c. Similarly, block 242 is
7 contained in separator 254, so separator 254 is split into two separators 256 and
8 258 as illustrated in Fig. 5d. Furthermore, block 244 is contained in separator 258,
9 so separator 258 is split into two separators 260 and 262 as illustrated in Fig. 5e.

10 When visual block 246 is analyzed, it is determined that visual block 246
11 crosses separator 256 and also covers separator 260. So, the parameters of
12 separator 256 are updated to reflect a smaller region, and separator 260 is removed
13 from the separator list as illustrated in Fig. 5f.

14 Additionally, each of the separators detected in process 200 is assigned a
15 weight. The weight of a separator is an indication of how visible the separator is
16 when the document is displayed (e.g., separators with higher weights are more
17 visible than separators with lower weights). The weight of a separator is assigned
18 based on various characteristics of the visual blocks on either side of the separator.
19 The weights assigned to separators are used for comparing the separators (e.g.,
20 their weights are compared). The weights assigned to the separators are typically
21 not used for other purposes, so the weights can have virtually any unit (so long as
22 it is consistent with or can be converted to a unit consistent with the units of
23 weights of other separators in the document).

24 A set of rules is applied to determine the weights of the separators. When
25 referring to blocks, the rules refer to the closest visual blocks on either side of the

separator (for example, in Fig. 5f, the closest visual blocks to separator 256 would be block 240 on one side and block 246 on the other side). The following is an example set of rules that can be applied to determine the weights of the separators:

- The greater the distance between blocks on different sides of the separator, the higher the weight of the separator. The distance can be, for example, the distance between the two closest edges of the two blocks. For example, for horizontal separators, if the distance is less than or equal to 10 pixels then set the weight of the separator to 15, if the distance is greater than 10 pixels but less than or equal to 20 pixels then set the weight of the separator to 25, and if the distance is greater than 20 pixels then set the weight of the separator to 35; for vertical separators, if the distance is less than or equal to 3 pixels then set the weight of the separator to 25, and if the distance is greater than 3 pixels then set the weight of the separator to 45.
- If a visual separator is at the same position as particular HTML tags (e.g., the <HR> HTML tag), the weight of the separator is made higher. For example, set the weight of the separator to 50 if the visual separator is at the same position as an <HR> HTML tag.
- For horizontal separators, if the font properties (e.g., font size and font weight) of blocks on different sides of the separator are different, the weight of the separator will be changed. This rule may apply only if, for each of the blocks, all of the text in the block has the same properties, or if at least a threshold amount of the text in the block has the same properties, etc. Additionally, the weight of the separator will be increased if the font size in the block before the

separator is smaller than the font size in the block after the separator. For example, if the font size used in the block before the separator (e.g., above a horizontal separator) is larger than the font size used in the block after the separator (e.g., below a horizontal separator), then set the weight of the separator based on the font size used in the block before the separator (e.g., set the weight to 9 if the font size is greater than or equal to 36 point, set the weight to 8 if the font size is greater than or equal to 24 point but less than 36 point, set the weight to 7 if the font size is greater than or equal to 18 point but less than 24 point, set the weight to 6 if the font size is greater than or equal to 14 point but less than 18 point, set the weight to 5 if the font size is greater than 12 point but less than 14 point, and set the weight to zero if the font size is less than or equal to 12 point). However, if the font size used in the block before the separator is smaller than the font size used in the block after the separator, then set the weight of the separator based on the font size used in the block after the separator (e.g., set the weight to 10 if the font size is greater than or equal to 36 point, set the weight to 9 if the font size is greater than or equal to 24 point but less than 36 point, set the weight to 8 if the font size is greater than or equal to 18 point but less than 24 point, set the weight to 7 if the font size is greater than or equal to 14 point but less than 18 point, set the weight to 6 if the font size is greater than 12 point but less than 14 point, and set the weight to zero if the font size is less than or equal to 12 point).

1 If background colors of blocks on different sides of the separator are
2 different, then the weight of the separator will be increased. For
3 example, if the background colors on different sides of the separator
4 are different, then set the weight of the separator to 40.

- 5 • For horizontal separators, when the structures of the blocks beside
6 the separator are very similar, then the weight of the separator will
7 be decreased. For example, if both blocks beside the separator are
8 text blocks having the same font size and the same font weight, then
9 set the weight of the separator to zero.

10 If multiple ones of these rules apply to a particular situation, then one of the
11 weights set by these rules can be selected, or the weights set by these rules can be
12 combined. For example, the smallest (or alternatively the largest) of the assigned
13 weights may be used as the weight of the separator, the average of all the assigned
14 weights may be used as the weight of the separator, a weighted average of all the
15 assigned weights may be used as the weight of the separator, and so forth.

16 17 Content Structure Construction

18 Content structure construction generates a content structure for the
19 document based on the blocks identified in the visual block identification as well
20 as the separators detected in the visual separator detection. Content structure
21 construction (e.g., in act 148 of Fig. 2 or implemented by content structure
22 constructor 108 of Fig. 1) can be performed as follows.

23 The content structure is a set of one or more portions of the document that
24 represents the semantic content of the document. In certain embodiments, the
25 content structure is a hierarchical tree structure of semantic content.

Fig. 6 is a flowchart illustrating an example process 280 for performing content structure construction. Process 280 illustrates act 148 of Fig. 2 in additional detail. Process 280 may be performed in software, firmware, hardware, or combinations thereof.

Initially, one or more virtual blocks are generated based on the detected separators and the identified visual blocks (act 282). The virtual blocks are generated by analyzing the detected separators, starting with those separators having the lowest weight. The blocks on either side of the detected separators are merged into a single virtual block. This merging continues with the separators having the next lowest weight, and continues until separators with a maximum weight are met (this maximum weight may be defined as the largest weight calculated by the visual separator detection discussed above for this particular set of identified visual blocks, or alternatively may be some pre-defined or pre-programmed value).

Once the virtual blocks are generated, a degree of coherence value is determined for each of the virtual blocks (act 284). The degree of coherence value is calculated based on the weight of the separator between the two blocks that were merged to generate the virtual block (this separator may be between two visual blocks or between two virtual blocks). In certain embodiments, the degree of coherence value for a virtual block is determined according to the following example rules. In some of these rules, two variables WEIGHT_TMP_TITLE2 and WEIGHT_TMP_TITLE1 are used. The values of the WEIGHT_TMP_TITLE2 and WEIGHT_TMP_TITLE1 variables are determined after all the separators are detected (e.g., after process 200 of Fig. 4 is finished). All of the separators having text blocks on both sides are sorted by weight. The highest of these weights is the

1 value assigned to the WEIGHT_TMP_TITLE1 variable, and the second highest of
2 these weights is the value assigned to the WEIGHT_TMP_TITLE2 variable. If
3 there are no such separators, then both of the WEIGHT_TMP_TITLE1 and
4 WEIGHT_TMP_TITLE2 variables are set to the value -1. If there is only one
5 such separator, then the weight of that one separator is used as the value of
6 WEIGHT_TMP_TITLE1, and the value -1 is used as the value of
7 WEIGHT_TMP_TITLE2. These example rules are:

- 8 • if the weight of the selector between the two blocks being merged is
9 less than or equal to zero, then set the degree of coherence value to
10 10;
- 11 • if the weight of the selector between the two blocks being merged is
12 less than or equal to WEIGHT_TMP_TITLE2, then set the degree of
13 coherence value to 7;
- 14 • if the weight of the selector between the two blocks being merged is
15 less than or equal to WEIGHT_TMP_TITLE1, then set the degree of
16 coherence value to 6;
- 17 • if the weight of the selector between the two blocks being merged is
18 less than or equal to 9, then set the degree of coherence value to 5;
- 19 • if the weight of the selector between the two blocks being merged is
20 less than or equal to 20, then set the degree of coherence value to 5;
- 21 • if the weight of the selector between the two blocks being merged is
22 less than or equal to 40, then set the degree of coherence value to 4;
- 23 • if the weight of the selector between the two blocks being merged is
24 less than or equal to 50, then set the degree of coherence value to 2;
- 25

1 if none of the above rules are satisfied, then set the degree of
2 coherence value to 1.

3 The content structure is then generated (act 286). The content structure is
4 generated based at least in part on the granularity requirement and which virtual
5 blocks, if any, satisfy the granularity requirement. As discussed above, each leaf
6 node of the content structure is analyzed to determine whether the granularity
7 requirement is satisfied. In certain embodiments, a permitted degree of coherence
8 (PDoC) value is defined, and, in order to satisfy the granularity requirement, each
9 leaf node of the content structure is required to have a DoC value greater than (or
10 alternatively greater than or equal to) the PDoC value. The PDoC value can be a
11 pre-defined value (e.g., determined empirically by a system designer). Any of a
12 range of values for the PDoC value can be used, with greater values typically
13 resulting in content structures with more but smaller visual blocks. An example
14 range of PDoC values is from 6 to 9.

15 The content structure generated in act 286 can include virtual blocks and/or
16 visual blocks. The DoC value for each virtual block identified in act 282 is
17 compared to the PDoC value. For each virtual block having a DoC value greater
18 than the PDoC value, the children of that virtual block are not output as blocks in
19 the content structure.

20 The content structure is then output (act 288). The content structure can be
21 output in any of a variety of formats, and in certain embodiments is output in a
22 hierarchical tree format, with the nodes of the tree representing virtual blocks
23 and/or visual blocks.

24 Fig. 7 illustrates an example of generating virtual blocks and a content
25 structure tree. In Fig. 7, six visual blocks 300, 302, 304, 306, 308, and 310 are

1 shown, having been identified by the visual block identification. Additionally,
2 separators 312, 314, 316, 318, and 320 between the blocks are illustrated, with the
3 thicker-lined separators (separators 314 and 318) being larger weights than the
4 thinner-lined separators (separators 312, 316, and 320).

5 Each of the visual blocks 300 – 310 has a corresponding content structure
6 block in the content structure tree 322. Additionally, a virtual block 324 is
7 generated by merging blocks 300 and 302, virtual block 324 having a DoC value
8 based on the weight of separator 312. A virtual block 326 is generated by merging
9 blocks 304 and 306, virtual block 326 having a DoC value based on the weight of
10 separator 316. A virtual block 328 is generated by merging blocks 308 and 310,
11 virtual block 328 having a DoC value based on the weight of separator 320.
12 Additionally, a virtual block 330 is generated by merging virtual blocks 324, 326,
13 and 328. The weights of separators 314 and 318 are the same, so the DoC value
14 for block 330 can be based on the weight of either separator 314 or 318.

15 The content structure output based on the virtual blocks illustrated in Fig. 7
16 depends on the DoC values of the virtual blocks 324, 326, 328, and 330. For
17 example, if the DoC value of virtual block 324 is greater than the PDoC value,
18 then virtual block 324 is output as a block in the content structure but the
19 individual visual blocks 300 and 302 are not output as blocks in the content
20 structure. However, if the DoC value of virtual block 324 is not greater than the
21 PDoC value, then virtual block 324 is not output as a block in the content structure
22 but the individual visual blocks 300 and 302 are output as blocks of the content
23 structure. By way of another example, if the DoC value of virtual block 330 is
24 greater than the PDoC value, then virtual block 330 is output as a block in the
25 content structure but the individual virtual blocks 324, 326, and 328 are not output

1 as blocks in the content structure, and the individual visual blocks 300, 302, 304,
2 306, 308, and 310 are not output as blocks in the content structure.

3 4 Document Retrieval

5 Document retrieval techniques can be employed using the vision-based
6 document segmentation described herein. Such document retrieval techniques
7 may rely on the vision-based document segmentation alone, or alternatively in
8 combination with more traditional search techniques

9 Fig. 8 is a flowchart illustrating an example process 360 for performing
10 document retrieval using the vision-based document segmentation described
11 herein. Process 360 may be implemented, for example, by document retrieval
12 module 110 of Fig. 1. Process 360 may be performed in software, firmware,
13 hardware, or combinations thereof. Process 360 can be used for retrieving any of
14 a variety of different types of documents, including, for example, web pages (e.g.,
15 available over the Internet and/or an intranet), papers written by individuals,
16 abstracts or summaries, and so forth.

17 Initially, a query is received (act 362). Typically, the query is input by a
18 user, although alternatively the query may be received from some other source
19 (e.g., a remote device, a software application, etc.). A database of blocks obtained
20 from the vision-based document segmentation technique discussed herein is
21 accessed (act 364). Typically, the vision-based document segmentation process
22 will have been previously performed and the resulting content structure (including
23 visual blocks and/or virtual blocks) already stored in a database or some other
24 accessible location. Alternatively, the vision-based document segmentation
25

1 process may be performed on one or more documents in response to receipt of the
2 query in act 362.

3 Once accessed, the blocks of the content structure are ranked according to
4 how well they match the query criteria (act 366). The visual blocks and/or the
5 virtual blocks can be ranked in act 366. The query received in act 362 includes
6 one or more search terms, also referred to as query criteria or query terms. In
7 certain embodiments, only those blocks that include at least one of the search
8 terms (or alternatively only those blocks that include all of the search terms) are
9 ranked in act 366. Alternatively, other limits may be imposed on the number of
10 blocks that are ranked (e.g., process 360 may be configured to rank only 50 or 100
11 blocks). The rankings based on the blocks can be generated in any of a variety of
12 different manners. For example, the rankings can be based on one or more of:
13 how many of the search terms are included in the block, the location of the various
14 search terms in the block, how frequently the search terms occur in the block, and
15 so forth.

16 Document rankings are then generated based on the block rankings (act
17 368). The document rankings can be generated in any of a variety of manners. In
18 certain embodiments, the rank of the highest ranking block from the document is
19 used as the ranking for the document. Alternatively, the rankings of all the blocks
20 in the document may be combined (e.g., by generating an average of the rankings
21 of all the blocks, by generating a weighted average of the rankings of all the
22 blocks, etc.) to generate the ranking for the document.

23 The document rankings are then returned (act 370), e.g., to the requester.
24 The document rankings can be returned in a variety of different manners, such as
25 an identifier (e.g., title or uniform resource locator (URL)) and numeric ranking

1 for each document, a display of an identifier of each document in order according
2 to their rankings (e.g., without displaying an explicit numeric ranking for each
3 document), excerpts from the documents that include the search criteria, the actual
4 documents returned in an order according to their rank, and so forth.

5 It is to be appreciated that various modifications can be made to process
6 360. For example, in act 366 not all of the accessible documents may be ranked or
7 searched. By way of another example, blocks of the documents may be returned
8 in act 370 rather than the entire documents.

9 Fig. 9 is a flowchart illustrating another example process 400 for
10 performing document retrieval using the vision-based document segmentation
11 described herein. Process 400 may be implemented, for example, by document
12 retrieval module 110 of Fig. 1. Process 400 may be performed in software,
13 firmware, hardware, or combinations thereof. Analogous to process 360 of Fig. 8,
14 process 400 can be used for retrieving any of a variety of different types of
15 documents, including, for example, web pages (e.g., available over the Internet
16 and/or an intranet), papers written by individuals, abstracts or summaries, and so
17 forth.

18 Process 400 is similar to process 360, however the vision-based document
19 segmentation is not performed on all of the possible documents. Rather, an initial
20 search is performed, and the vision-based document segmentation process is
21 performed using the documents from the initial search.

22 Initially, a query is received (act 402). Analogous to process 360, the query
23 may be a user query or from some other source, and the query includes one or
24 more search terms. An initial document retrieval is performed on the document
25 set, and the documents satisfying the query criteria are ranked (act 404). This

1 initial retrieval can be performed in any of a variety of manners. An example of
2 such an initial retrieval process in act 404 is described in Robertson, S.E.,
3 "Overview of the Okapi Projects", Journal of Documentation, Vol. 53, No. 1,
4 1997, pp. 3-7. Any of a variety of conventional web search engines can also be
5 used in act 404.

6 Process 400 then proceeds with performing the vision-based document
7 segmentation process on the documents retrieved in the initial retrieval (act 406).
8 Block retrieval is then performed using the query criteria (search terms) and the
9 blocks obtained from the vision-based document segmentation (act 408),
10 analogous to act 366 of Fig. 8. Refined document rankings are then generated
11 based on the block ranks (act 410), analogous to the discussion above in act 366 of
12 Fig. 8.

13 Final document rankings are then generated by combining the initial
14 document rankings from act 404 and the refined document rankings from act 410
15 (act 412). This combining can be performed in any of a variety of manners. For
16 example, the rankings from acts 404 and 410 may be averaged together, or a
17 weighted average for the rankings from acts 404 and 410 may be used.
18 Alternatively, the highest of the rankings from acts 404 and 410 for a particular
19 document may be used as the ranking for that document. In yet another
20 alternative, the ranking from act 410 may be used in place of the ranking from act
21 404.

22 Once the final document rankings are generated, the document rankings are
23 returned (act 414), analogous to act 370 of Fig. 8.

24 Fig. 10 is a flowchart illustrating another example process 440 for
25 performing document retrieval using the vision-based document segmentation

1 described herein. Process 440 may be implemented, for example, by document
2 retrieval module 110 of Fig. 1. Process 440 may be performed in software,
3 firmware, hardware, or combinations thereof. Analogous to process 360 of Fig. 8,
4 process 440 can be used for retrieving any of a variety of different types of
5 documents, including, for example, web pages (e.g., available over the Internet
6 and/or an intranet), papers written by individuals, abstracts or summaries, and so
7 forth.

8 Process 440 is similar to process 400 of Fig. 9, however the vision-based
9 document segmentation is used to expand the query criteria.

10 Initially, a query is received (act 442), analogous to act 402 of Fig. 9. An
11 initial document retrieval is performed on the document set, and the documents
12 satisfying the query criteria are ranked (act 444), analogous to act 404 of Fig. 9.
13 The vision-based document segmentation process is then performed on the
14 documents retrieved in the initial retrieval (act 446), analogous to act 406 of
15 Fig. 9.

16 However, rather than performing block retrieval as is done in Fig. 9, the
17 blocks of the content structure obtained from the vision-based document
18 segmentation process, along with the query criteria, are used to perform query
19 expansion (act 448). The query expansion can be performed in any of a variety of
20 manners.

21 In certain embodiments, the query expansion is performed by using the top-
22 ranking blocks for expansion term selection (e.g., the ten blocks having the highest
23 rankings, or the 10% of the blocks having the highest rankings). The blocks are
24 ranked according to how well they satisfy the original query terms (analogous to
25 act 366 of Fig. 8). The expansion terms are selected in any of a variety of known

1 manners, however, unlike traditional query expansion, the expansion terms are
2 selected based on the blocks obtained from the vision-based document
3 segmentation process. For example, all of the terms (except for the original query
4 terms) in the blocks obtained from the vision-based document segmentation
5 process are weighted according to the following term selection value *TSV*:

$$6 \quad TSV = w^{(1)} \times \frac{r}{R}$$

7 where r represents the number of blocks which contain the term, R represents the
8 total number of blocks obtained from the vision-based document segmentation
9 process, and $w^{(1)}$ represents the Robertson/Spark Jones weight of T in Q , where Q
10 represents the original query (received in act 442) and T represents the search
11 terms in the original query.

12 Given these weighted expansion terms, a number of these terms are
13 selected to be used as the expansion terms. Various numbers of these terms can be
14 selected, and in certain embodiments the top ten terms are selected as the
15 expansion terms (that is, the ten terms having the largest term selection values
16 *TSV*).

17 A final retrieval is then performed based on the expanded query criteria and
18 the document set, and the documents satisfying the expanded query criteria are
19 ranked (act 450). Act 450 is performed analogous to act 444, except that the
20 expanded query is used rather than the received criteria. The expanded query
21 includes the original query terms as well as the expansion terms. In certain
22 embodiments, the terms in the expanded query are weighted. The terms can be
23 weighted in a variety of manners to give a preference to the original query terms
24 when determining which documents match the query terms and/or when ranking
25 the documents. For example, the terms for the expanded query can be weighted as

1 follows. For original terms (those terms received as part of the query in act 442),
2 the new weight of the terms is $tf \times 2$, where tf represents the term's frequency in
3 the query. For each expansion term, the weight of the term is set as follows:

$$4 \quad \frac{1 - (n - 1)}{m}$$

5 where n represents the *TSV* rank value of the term, and m is the number of
6 expansion terms (e.g., ten in certain embodiments).

7 The document rankings obtained from the final retrieval in act 450 are then
8 returned (act 452), analogous to act 414 of Fig. 9.

9 Additionally, in certain embodiments the document retrieval discussed
10 above with reference to Fig. 8, Fig. 9, or Fig. 10 may employ a combined
11 document segmentation approach which combines the visual-based document
12 segmentation process discussed above with a fixed length approach. In this
13 combined document segmentation approach, visual blocks are obtained as
14 discussed above (e.g., with reference to Fig. 2). Given these visual blocks,
15 overlapped windows are used to divide at least some of the blocks into smaller
16 units. The first window begins from the first word of the first visual block and
17 subsequent windows overlap preceding windows by a certain amount (e.g., one-
18 half) until the end of the block. The portions of the visual block corresponding to
19 these windows are output as the final block. For visual blocks that are smaller
20 than the length of the window (the window typically being a fixed value), the
21 visual blocks are output as final blocks without further partition. However, by
22 partitioning larger blocks into smaller blocks that are the size of the window, the
23 variance in lengths of the various blocks is reduced.

24 Using this combined document segmentation approach, the final blocks
25 output by the combined document segmentation approach are used in place of the

1 visual and/or virtual blocks of the content structure during the document retrieval
2 process.

4 Example Environment

5 Fig. 11 illustrates an example of a general computer environment 500,
6 which can be used to implement the techniques described herein. The computer
7 environment 500 is only one example of a computing environment and is not
8 intended to suggest any limitation as to the scope of use or functionality of the
9 computer and network architectures. Neither should the computer environment
10 500 be interpreted as having any dependency or requirement relating to any one or
11 combination of components illustrated in the example computer environment 500.

12 Computer environment 500 includes a general-purpose computing device in
13 the form of a computer 502. Computer 502 can implement, for example, visual
14 block extractor 104, visual separator detector 106, content structure constructor
15 108, and/or document retrieval module 110 of Fig. 1. The components of
16 computer 502 can include, but are not limited to, one or more processors or
17 processing units 504, a system memory 506, and a system bus 508 that couples
18 various system components including the processor 504 to the system memory
19 506.

20 The system bus 508 represents one or more of any of several types of bus
21 structures, including a memory bus or memory controller, a peripheral bus, an
22 accelerated graphics port, and a processor or local bus using any of a variety of
23 bus architectures. By way of example, such architectures can include an Industry
24 Standard Architecture (ISA) bus, a Micro Channel Architecture (MCA) bus, an
25 Enhanced ISA (EISA) bus, a Video Electronics Standards Association (VESA)

1 local bus, and a Peripheral Component Interconnects (PCI) bus also known as a
2 Mezzanine bus.

3 Computer 502 typically includes a variety of computer readable media.
4 Such media can be any available media that is accessible by computer 502 and
5 includes both volatile and non-volatile media, removable and non-removable
6 media.

7 The system memory 506 includes computer readable media in the form of
8 volatile memory, such as random access memory (RAM) 510, and/or non-volatile
9 memory, such as read only memory (ROM) 512. A basic input/output system
10 (BIOS) 514, containing the basic routines that help to transfer information
11 between elements within computer 502, such as during start-up, is stored in ROM
12 512. RAM 510 typically contains data and/or program modules that are
13 immediately accessible to and/or presently operated on by the processing unit 504.

14 Computer 502 may also include other removable/non-removable,
15 volatile/non-volatile computer storage media. By way of example, Fig. 11
16 illustrates a hard disk drive 516 for reading from and writing to a non-removable,
17 non-volatile magnetic media (not shown), a magnetic disk drive 518 for reading
18 from and writing to a removable, non-volatile magnetic disk 520 (e.g., a "floppy
19 disk"), and an optical disk drive 522 for reading from and/or writing to a
20 removable, non-volatile optical disk 524 such as a CD-ROM, DVD-ROM, or other
21 optical media. The hard disk drive 516, magnetic disk drive 518, and optical disk
22 drive 522 are each connected to the system bus 508 by one or more data media
23 interfaces 526. Alternatively, the hard disk drive 516, magnetic disk drive 518,
24 and optical disk drive 522 can be connected to the system bus 508 by one or more
25 interfaces (not shown).

1 The disk drives and their associated computer-readable media provide non-
2 volatile storage of computer readable instructions, data structures, program
3 modules, and other data for computer 502. Although the example illustrates a hard
4 disk 516, a removable magnetic disk 520, and a removable optical disk 524, it is to
5 be appreciated that other types of computer readable media which can store data
6 that is accessible by a computer, such as magnetic cassettes or other magnetic
7 storage devices, flash memory cards, CD-ROM, digital versatile disks (DVD) or
8 other optical storage, random access memories (RAM), read only memories
9 (ROM), electrically erasable programmable read-only memory (EEPROM), and
10 the like, can also be utilized to implement the example computing system and
11 environment.

12 Any number of program modules can be stored on the hard disk 516,
13 magnetic disk 520, optical disk 524, ROM 512, and/or RAM 510, including by
14 way of example, an operating system 526, one or more application programs 528,
15 other program modules 530, and program data 532. Each of such operating
16 system 526, one or more application programs 528, other program modules 530,
17 and program data 532 (or some combination thereof) may implement all or part of
18 the resident components that support the distributed file system.

19 A user can enter commands and information into computer 502 via input
20 devices such as a keyboard 534 and a pointing device 536 (e.g., a "mouse").
21 Other input devices 538 (not shown specifically) may include a microphone,
22 joystick, game pad, satellite dish, serial port, scanner, and/or the like. These and
23 other input devices are connected to the processing unit 504 via input/output
24 interfaces 540 that are coupled to the system bus 508, but may be connected by
25

1 other interface and bus structures, such as a parallel port, game port, or a universal
2 serial bus (USB).

3 A monitor 542 or other type of display device can also be connected to the
4 system bus 508 via an interface, such as a video adapter 544. In addition to the
5 monitor 542, other output peripheral devices can include components such as
6 speakers (not shown) and a printer 546 which can be connected to computer 502
7 via the input/output interfaces 540.

8 Computer 502 can operate in a networked environment using logical
9 connections to one or more remote computers, such as a remote computing device
10 548. By way of example, the remote computing device 548 can be a personal
11 computer, portable computer, a server, a router, a network computer, a peer device
12 or other common network node, and the like. The remote computing device 548 is
13 illustrated as a portable computer that can include many or all of the elements and
14 features described herein relative to computer 502.

15 Logical connections between computer 502 and the remote computer 548
16 are depicted as a local area network (LAN) 550 and a general wide area network
17 (WAN) 552. Such networking environments are commonplace in offices,
18 enterprise-wide computer networks, intranets, and the Internet.

19 When implemented in a LAN networking environment, the computer 502 is
20 connected to a local network 550 via a network interface or adapter 554. When
21 implemented in a WAN networking environment, the computer 502 typically
22 includes a modem 556 or other means for establishing communications over the
23 wide network 552. The modem 556, which can be internal or external to computer
24 502, can be connected to the system bus 508 via the input/output interfaces 540 or
25 other appropriate mechanisms. It is to be appreciated that the illustrated network

1 connections are examples and that other means of establishing communication
2 link(s) between the computers 502 and 548 can be employed.

3 In a networked environment, such as that illustrated with computing
4 environment 500, program modules depicted relative to the computer 502, or
5 portions thereof, may be stored in a remote memory storage device. By way of
6 example, remote application programs 558 reside on a memory device of remote
7 computer 548. For purposes of illustration, application programs and other
8 executable program components such as the operating system are illustrated herein
9 as discrete blocks, although it is recognized that such programs and components
10 reside at various times in different storage components of the computing device
11 502, and are executed by the data processor(s) of the computer.

12 13 Conclusion

14 Various modules and techniques may be described herein in the general
15 context of computer-executable instructions, such as program modules, executed
16 by one or more computers or other devices. Generally, program modules include
17 routines, programs, objects, components, data structures, etc. that perform
18 particular tasks or implement particular abstract data types. Typically, the
19 functionality of the program modules may be combined or distributed as desired in
20 various embodiments.

21 An implementation of these modules and techniques may be stored on or
22 transmitted across some form of computer readable media. Computer readable
23 media can be any available media that can be accessed by a computer. By way of
24 example, and not limitation, computer readable media may comprise "computer
25 storage media" and "communications media."

1 “Computer storage media” includes volatile and non-volatile, removable
2 and non-removable media implemented in any method or technology for storage
3 of information such as computer readable instructions, data structures, program
4 modules, or other data. Computer storage media includes, but is not limited to,
5 RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM,
6 digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic
7 tape, magnetic disk storage or other magnetic storage devices, or any other
8 medium which can be used to store the desired information and which can be
9 accessed by a computer.

10 “Communication media” typically embodies computer readable
11 instructions, data structures, program modules, or other data in a modulated data
12 signal, such as carrier wave or other transport mechanism. Communication media
13 also includes any information delivery media. The term “modulated data signal”
14 means a signal that has one or more of its characteristics set or changed in such a
15 manner as to encode information in the signal. By way of example, and not
16 limitation, communication media includes wired media such as a wired network or
17 direct-wired connection, and wireless media such as acoustic, RF, infrared, and
18 other wireless media. Combinations of any of the above are also included within
19 the scope of computer readable media.

20 Various flowcharts are described herein and illustrated in the
21 accompanying Figures. The ordering of acts in these flowcharts are examples
22 only – these orderings can be changed so that the acts are performed in different
23 orders and/or concurrently.

24 Additionally, many specific numeric examples are given herein (e.g.,
25 particular threshold values, particular sizes, particular weights, etc.). These

1 specific numeric examples are only examples, and other values can alternatively
2 be used.

3 Although the description above uses language that is specific to structural
4 features and/or methodological acts, it is to be understood that the invention
5 defined in the appended claims is not limited to the specific features or acts
6 described. Rather, the specific features and acts are disclosed as exemplary forms
7 of implementing the invention.